University of Castilla-La Mancha

A publication of the

Computing Systems Department

A Taxonomy of Proposals for Network QoS in Grid Systems

by

Agustín Caminero, Blanca Caminero, Carmen Carrión

Technical Report **#DIAB-07-01-1** August, 2007

This work has been jointly supported by the Spanish MEC and European Commission FEDER funds under grants "Consolider Ingenio-2010 CSD2006-00046" and "TIN2006-15516-C04-02"; by JCCM under grants PBC-05-007-01, PBC-05-005-01 and José Castillejo.

DEPARTAMENTO DE SISTEMAS INFORMï¿½TICOS
ESCUELA POLITÉCNICA SUPERIOR
UNIVERSIDAD DE CASTILLA-LA MANCHA
Campus Universitario s/n
Albacete - 02071 - Spain
Phone +34.967.599200, Fax +34.967.599224

A Taxonomy of Proposals for Network QoS in Grid Systems

Agustín Caminero, Blanca Caminero, Carmen Carrión
Computing Systems Department. Escuela Politécnica Superior
Universidad de Castilla-La Mancha. 02071 - Albacete, SPAIN
{agustin, blanca, carmen}@dsi.uclm.es

August 16, 2007

Abstract

Grid systems are highly variable environments, made of a series of independent organizations that share their resources, creating what is known as *virtual organization, VO*. This variability makes network quality of service (*QoS*) highly desirable, though it is very complex to achieve due to the large scale of interconnected networks. The provision of network QoS in a Grid environment is the topic of interest of this work. In this paper, a taxonomy that characterizes and classifies various approaches for providing network QoS in Grid systems is presented. Also, the main proposals, developed by various world-wide projects, are surveyed to demonstrate the comprehensiveness of the taxonomy.

1 Introduction

Grid computing has emerged as the next-generation parallel and distributed computing methodology that aggregates dispersed heterogeneous resources for solving various kinds of large-scale parallel applications in science, engineering and commerce [19].

They are highly variable environments, made of a series of independent organizations that share their resources, creating what is known as *virtual organization, VO* [21]. In Grid systems, each organization or administrative domain connected to a Grid keeps its independence and autonomy, and this is an essential issue for the creation of a real Grid [18].

Grid computing technologies are increasingly being used to aggregate computing resources that are geographically distributed across different locations. Large-scaled networks, usually the Internet, are being used to connect these computing resources, and thus serve as a fundamental component of Grid computing. Since these Grid resources are connected over a shared infrastructure, it is essential that its effect is considered in the performance of the jobs executed in the Grid.

Because of the nature of a Grid, users may want to transfer large amount of data between different locations, across wide area networks. There are many users within the Grid community with such requirements, e.g. astronomers [1] or high-energy physicists [2]. These users, among others, may also want a kind of "guarantee" on the performance that their data transfer will get. So, they will need to ask the network for a reservation in order to achieve such guarantee. For these users, some mechanisms are required to enable QoS services. Application level mechanisms are needed too in order to use such services.

Some proposals have been made to provide Grid systems with network QoS [33] [8] [6] [11]. In this paper a taxonomy is shown in which these proposals have been classified based on different aspects and features.

This paper is organized as follows: in Section 2 a taxonomy of proposals to provide network QoS in a Grid system is presented. This taxonomy has been made taking into account a number of features, namely application model, Grid information service,

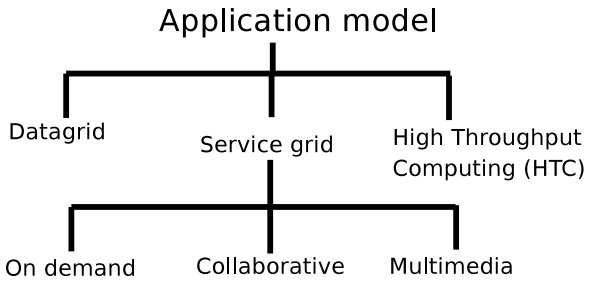

Figure 1: Classification of proposals based on the application model.

type of resources, scheduling, and communication mechanisms. After that, Section 3 gives an insight into those proposals. Finally, Section 4 concludes the paper.

2 Taxonomy

The proposed taxonomy provides a classification of the proposals for network QoS in Grid systems into categories based on aspects and properties. The categories found will be explained in this section, and are depicted in Figures 1, 2, 3, 4 and 5. Figure 1 shows a classification based on the application model, Figure 2 classifyes the proposals based on the information service and Figure 3 based on the types of resources each proposal can deal with. Figure 4 depicts different scheduling methods. Finally, Figure 5 sketches the communication mechanisms proposals can use. They are all revised in this section.

2.1 Application model

The stress imposed on the network depends on the type of application users want to run. For example, applications requiring a heavy network input/output are more sensitive to network performance than other aplications. Thus, one of the parameters to take into account is the *application model*.

The classification shown here has been made taking into account those appearing in [27] and [41], and is depicted in Figure 1.

- Data Grid: Data Grids primarily deal with providing services and infrastructure for distributed data-intensive applications that need to access, transfer and modify massive datasets stored in distributed storage resources. Data Grids aim to combine high-end computing technologies with high-performance networking and wide-area storage management techniques [41]. Typical applications for these systems include special purpose data mining that correlates information from multiple different data sources. The data Grid initiatives, European DataGrid Project [24] and Globus [13], are working on developing large-scale data organization, catalog, management, and access technologies. As this kind of applications requires the transmission of large datasets, the network has a high significance on the performance of applications. Thus, this class has been considered more relevant for the purposes of the authors' work than the other classes found.

- High Throughput Computing (HTC) Grid: For many research and engineering projects, the quality of the research or the product is heavily dependent upon the quantity of computing cycles available. It is not uncommon to find problems that require weeks or months of computation to solve. Scientists and engineers engaged in this sort of work need a computing environment that delivers large amounts of computational power over a long period of time. Such an environment is called a *High-Throughput Computing* (HTC) environment. A high throughput Grid increases the completion rate of a stream of jobs and are well suited for 'parameter sweep' type applications such as Monte Carlo simulations [27] [3] [10].

- Service Grid: The service Grid category is for systems that provide services that are not provided by any single machine [27]. In this category several sub-

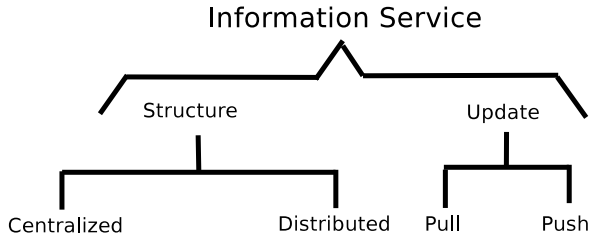

Figure 2: Classification of proposals based on the information service.

categories have been identified, namely on-demand, collaborative, and multimedia Grid systems [27]:

- On demand: In this case, new resources can be dynamically aggregated to provide new services. A data visualization workbench that allows a scientist to dynamically increase the fidelity of a simulation by allocating more machines to a simulation would be an example of an on-demand Grid system.
- Collaborative: Connects users and applications into collaborative workgroups. These systems enable real time interaction between humans and applications via a virtual workspace.
- Multimedia: Provides an infrastructure for real-time multimedia applications. This requires supporting QoS across multiple different machines whereas a multimedia application on a single dedicated machine may be deployed without QoS [30].

2.2 Grid Information Service

As network QoS proposals have to perform reservation and allocation of resources, a Grid Information Service (GIS) should also be included in this taxonomy.

Grid Information Service (GIS) is a software component, whether singular or distributed, that maintains information about people, software, services, and hardware

that participate in a computational Grid, and makes that information available upon request. A GIS can also provide binding, discovery, lookup, and data protection. While directory services are generally focused at the network level, that is, mapping a host-name to an IP address, or a location transparent name to its host-specific equivalent, a GIS must represent a richer set of entities having richer relationships between them and more dynamic information. For instance, whereas a machine's IP address is quite stable, the current average CPU load over multiple CPUs in an SMP can be as dynamic as an information service will allow [31].

As many of the features in a Grid system, the provision of network QoS depends on a GIS. A GIS must provide information regarding the resources currently available for use, and their current level of use. Also, it will have to keep this information up-to-date.

The accuracy of the information that the GIS provides will determine the performance of the network QoS proposals. If the information is not up-to-date, users may submit their jobs to a resource that does not exist any more, thus leading the user to get a job failure. Also, users may submit their jobs to a resource whose links are more loaded than expected, thus increasing the network latencies suffered by jobs. So, the accuracy of the information is key in this topic.

Hence, for the sake of the completeness of this taxonomy, the GIS will be included as a category. A GIS can be classified based on the properties shown in Figure 2.

In principle, two different types of GIS structures can be distinguished: centralized (hierarchical) and distributed. *Centralized structures* are organized similar to the web of a spider. All the information flow is managed by a central entity that keeps record of all the resources available in the system. So, if the Grid extension increases, scalability problems may arise with this centralized architecture. *Distributed structures* solve the problem of scalability, at the expense of increasing the complexity of

the GIS. This is because communication and coordination among the different parts of the distributed structure should be kept.

Other parameter to be taken into account is the implementation of the GIS, this is, the way how the GIS keeps its information up-to-date. This also includes the detection of resource failures, among other things. There are two general methods for failure detection [22], i.e. *push* and *pull*.

The *push* method uses some form of "heartbeat" messages to renew a soft-state availability registration [20]. Each monitored resource periodically sends a message to a central server indicating its availability. Missing a heartbeat after a certain time interval (timeout) T indicates that this resource has failed. Although this method is robust when the central server is running on a highly available system, it is inextricably convolving network failure and host failure. That is, a missing heartbeat or set of heartbeats can either be interpreted as the failure in the monitored resource or the loss of network connection. A real implementation using the heartbeat method can be found at [38].

The *pull* method works by sending a message or a polling request to the monitored resources. On the reception of these messages, the resources will send the message back. When the sender receives the messages back, it will understand that the resource is alive. However, if T expires before the reply message comes back, it will understand that this resource is not available at this moment. Hence, the sender will keep an up-to-date list of available resources. The *pull* method has been implemented in the GridBus Broker [42] and the GridWay [25] metascheduler.

In order to carry out the pull method, the GIS should have a list of available resources. This means that when a resource connects to the Grid, it has to register itself at a GIS. This way, the GIS can carry out the pull method to keep its list of available resources up-to-date.

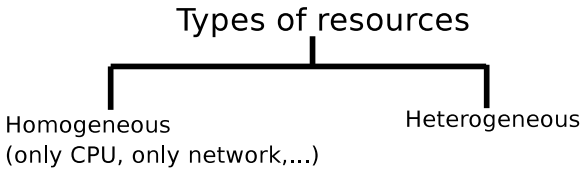

Figure 3: Classification of proposals based on types of resources.

Of course, for both methods, the information flow between resources and GIS can be used to send information regarding the current level of use of each resource. This way, the GIS can provide more efficient and useful information which can improve the performance received by users.

2.3 Type of resources

This classification has been made taking into account the resources each proposal can deal with. The classes found are depicted in Figure 3.

Some of the proposals have been specifically developed to work only with network resources, whilst others have been designed in a more general way to be able to provide QoS on a variety of resources, such as network, CPU, storage, ... This fact makes some of them more general tools, able to provide more complex QoS guarantees.

2.4 Scheduling

The scheduling method is another interesting parameter to take into account. This refers to the way that each proposal decides which computing resource will run each user's job. Proposals can be classified by means of the method used, the organization of the proposal, the optimization function and the reservation support. All of these categories will be explained in the following sections.

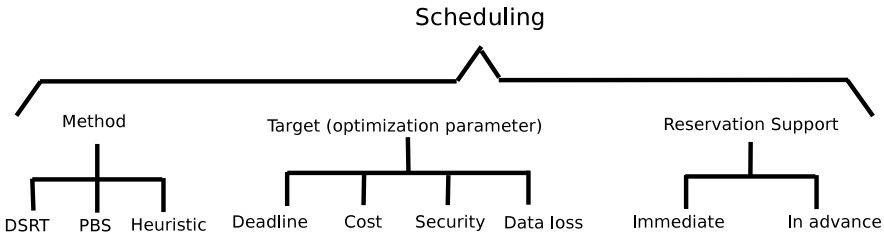

Figure 4: Scheduling.

2.4.1 Method

The existing proposals that provide network QoS in Grid, and include scheduling of users' jobs to computing resources, provide the scheduling by means of the following tools, namely DSRT [14] and PBS [40] [28].

The *Dynamic Soft Real Time Scheduler (DSRT)* is a user-level process that runs at a higher priority than the Unix scheduler and takes over the work of scheduling in order to provide soft real-time guarantees to applications. DSRT was built by a research group at the University of Illinois Urbana-Champagne and is described in detail in [14]. The DSRT system support multiple CPU service classes for the real time processes based on their processor usage pattern including periodic constant processing time class and periodic variable processing time class. It also provides the following features: (1) reservation and processing time guarantees for the service classes, (2) overrun protection and scheduling algorithm, and (3) system-initiated adaptation strategies. The other main feature of DSRT is its easy portability to any operating system with real time extensions because it is implemented purely in the user space without any modifications to the kernel.

DSRT uses Dynamic Earliest Deadline First scheduling algorithm. Earliest deadline first (ED) scheduling is a dynamic scheduling principle used in real-time operating systems. It places processes in a priority queue. Whenever a scheduling event occurs

(task finishes, new task released, etc.) the queue will be searched for the process closest to its deadline. This process will then be scheduled for execution next. The dynamic earliest deadline first algorithm sorts the processes according to their response deadline: the one having the nearest deadline will be executed first [35].

The *Portable Batch System (PBS)* [40] [28] is a batch job and computer system resource management package. It will accept batch jobs (shell scripts with control attributes), preserve and protect the job until it is run, run the job, and deliver output back to the submitter. PBS may be installed and configured to support jobs run on a single system, or many systems grouped together. Because of the flexibility of PBS, the systems may be grouped in many fashions.

PBS provides a separate process to schedule which jobs should be placed into execution [15]. This is a flexible mechanism by which you may implement a very wide variety of policies. In fact it is possible to implement a replacement Scheduler using the provided APIs which will enforce the desired policies. The configuration required for a Scheduler depends on the Scheduler itself. PBS comes with a FIFO Scheduler, which provides the ability to sort the jobs in several different ways, in addition to FIFO order. There is also the ability to sort on user and group priority. Mainly this Scheduler is intended to be a jumping off point for a real Scheduler to be written. A good amount of code has been written to make it easier to change and add to this Scheduler. As distributed, the FIFO Scheduler is configured with the following options [15]:

- The jobs within each queue are sorted by requested CPU time. The shortest job is places first.

- These system resources are checked to make sure they are not exceeded the memory requested and number of CPUs requested.

Apart from these techniques, some heuristic techniques can be developed which are aimed at providing an effective scheduling.

2.4.2 Optimization function

A job scheduling algorithm tries to minimize or maximize some form of optimization parameter [41]. The optimization parameter can vary depending on the requirements of the users and architecture of the distributed system that the algorithm is targeted at. Traditionally, some of the optimization parameters that have been used are the deadline, cost, security, and data loss.

Regarding the deadline, what the scheduling algorithm has to minimize is the makespan of the users' jobs, in order to meet their deadlines. Cost minimizing algorithms are aimed at getting the jobs executed spending as less budget as possible. Other optimization parameters are security and data loss, which are specially aimed at networks. Regarding data loss, this may happen in any network, so the user or application must have the ability to detect it and fix it.

2.4.3 Reservation support

In most Grid scheduling systems, submitted jobs are initially placed into a queue if there are no available resources [39]. Therefore, there is no guarantee as to when these jobs will be executed. This causes problems in parallel applications, where most of the jobs have dependencies among each other. Reservation is the process of requesting resources, so that the applications can be executed with little interference between each other. Common resources whose usage can be reserved or requested are CPUs, memory, disk space and network bandwidth.

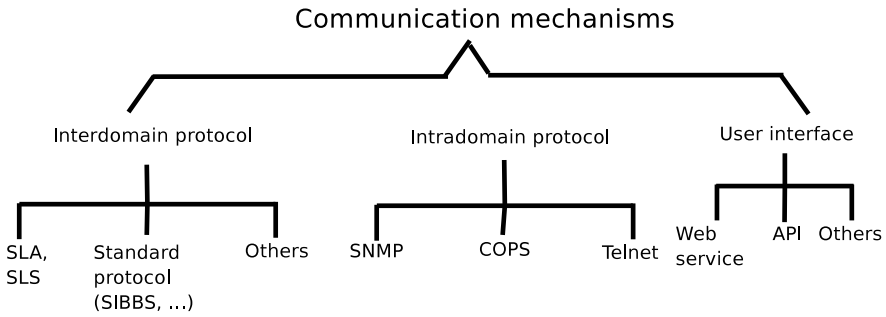

Figure 5: Communication mechanisms.

Reservations can be immediate or in advance. Advance Reservations (AR) are to use the resource at a specific time in the future [36]. Reservations for a Grid resource solve the above problem by allowing users to gain concurrent access to adequate resources for applications to be executed. Reservations also guarantee the availability of resources to users and applications at the required times [39].

2.5 Communication mechanisms

Any network QoS proposal must present a number of communication mechanisms, as they have to interact with other entities. Basically, there are three mechanisms needed [37], namely:

- Inter-domain protocol.
- Intra-domain protocol.
- User interface.

Each of them is used for communication with different entities, and will be explained in the following sections.

2.5.1 Inter-domain protocol

A common inter-domain interface must be available, allowing the QoS set-up of inter-domain traffic streams while hiding intra-domain characteristics from inter-network operations. This interface can be implemented by the definition of messages on top of protocols (or modification of protocols) such as the NSIS protocol suite, SIP, or protocols based on HTTP [29]. To allow easier parsing and more flexibility in setting security actions, such messages may be defined in a language like XML. The inter-domain interface should be independent from the specifics of the intra-domain control plane.

The DiffServ architecture [9] defines a *Service Level Agreement* (SLA) as a formal contractual agreement which contains issues such as network availability, payment agreements as well other legal and business related issues [26]. An SLA guarantees that traffic offered by a customer that meets the agreed conditions will be carried and delivered by the service provider. Depending on the contractual agreement, failure to provide the agreed service could result in some form of monetary or legal consequences. Not only does an SLA contain the technical forwarding behaviour that a certain traffic flow will receive, it could contain additional parameters like delay and access privileges.

Therefore an SLA is a partially technical document that is determined by network administrators and customers binded by legal obligations like any other contract.

Since an SLA contains non-technical information, it cannot be directly used by a bandwidth broker. An *Service Level Specification* (SLS) contains exclusively the technical details specified by an SLA. It is essentially a translation of a SLA into the appropriate information necessary to configure network devices. The SLS will dictate how traffic is dealt with within a DiffServ domain, from whether the flow is allowed access to that domain, to the forwarding behaviour it should receive within

that domain. Once a domain has agreed to honour the conditions set out in a SLS, it must be responsible for giving the guaranteed service to traffic specified in the SLS for the duration of the agreement. It is important to note that an SLS is not a reservation, it is rather a commitment to allow reservations based on the conditions found in the SLS [29].

Communication over a common inter-domain interface must be made based on a well-understood information model for SLSs. This model should allow the definition of different degrees of SLSs, from per-flow, more suitable for end-hosts or small networks, to per-aggregate, more suitable for large networks. It should also allow the identification of the SLS validity and a set of time periods over each the SLS must be available (activated), besides the information about the QoS characteristics. To allow for extra flexibility, the information model should be defined in an extendable language such as XML.

The inter-domain interface must allow network domains to offer, request/negotiate, and monitor agreements/SLSs [29]. Policy information specific to the requester, or other general policies must be checked to determine if the requested agreement can the accepted.

Each domain must ensure that the data packets it sends are in conformity with the established agreement or risk of having packets dropped. Packets should be re-marked from the internal PDB (Per Domain Behavior) identifier to the inter-domain SLS identifier if needed. At the provider side, packets may be re-marked from the SLS identifier to its internal PDB.

End-to-end signaling may be used to check the available guarantees in a chain of DiffServ domains. In some domains, this signaling may check that a suitable agreement is in place, in others the signaling may 'ask' for the 'activation' of a specific agreement

(assuming that agreements are already negotiated but they are associated to another time period, for instance), or it may ask for the negotiation of a specific agreement.

2.5.2 Intra-domain protocol

Any network QoS proposal must present a common intra-domain interface, similar to the one found in a DiffServ domain [29]. This is nececsary to allow network elements to coordinate their actions, independently of the degree of control distribution. This interface can be defined by messages sent over protocols such as COPS, SNMP, or the NSIS protocol suite [29]. Moreover, other proprietary solutions, such as Cisco's Modular QoS Command-line interface can be used. In fact, an abstract interface can be used with plugable modules able to tranlate general configuration specifications to the particular interface being used by each device.

Intra-domain operations must be supported by a list of policies and rules, as well as a group of databases. The latter should provide information about the state of the network, most likely collected by monitoring operations. Some intra-domain operations may also need to access databases related to inter-domain control, namely the set of established and available agreements.

2.5.3 User interface

Applications must state their QoS requirements in the form of a utility function and an adaptation policy [23]. The utility function expresses the desired application requirements with different levels of network bandwidth, while the adaptation policy determines how the applications' bandwidth allocation should vary as resource availability changes.

An uniform and intuitive user interface is essential to allow users to state their QoS requirements. Two different user interfaces are considered in this work (as shown in Figure 5): a web interface and an *Application Programming Interface* (API).

A user friendly web based client is available for users to access the proposal remotely to request for network resources. The web based interface with graphical user interface provides flexibility and ease to users to request and query the status of the network and perform reservations. Moreover, the network administrator can control and manage the entity through an interface [37]. The web interface is normally used to provide users with an straight-forward way to request network reservations, whereas an API is used to allow users' applications to provide QoS support to their users.

Applications can easily access the network QoS proposal through the API and request for resources on behalf of their users. Integrating this API into any resource manager like a software that manages the processing resources of a super computer enables that software to reserve the network resources also for the users without changing its basic structure.

3 Survey of network QoS proposals

A number of proposals have come out whose aim is the provision of network QoS in a Grid system. The best known is GARA. NRSE, G-QoSM, and GNB are other proposals aimed at providing network QoS in Grid systems. This section briefly describes these tools by highlighting their similarities and differences, and the categorization based on their features.

Proposal	GARA	NRSE	G-QoSM	GNB
Application model	Others	DataGrid	Others	DataGrid
Information service	Distributed	Any	Distributed	Any
Types of resources	A variety of resources (network, CPU, ...)	Only network	A variety of resources (network, CPU, ...)	Only network
Scheduling method	DSRT, PBS	-	DSRT	Heuristic
Network is considered at scheduling	No	-	No	Yes
Scheduling target	Deadline	-	Deadline	Deadline, data loss
Reservation support	Immediate and in advance	Immediate and in advance	Immediate and in advance	Immediate
Inter-domain protocol	Specific protocols	SLA	SLA, SIBBS	SLA
Intra-domain protocol	Proprietary solutions	Proprietary solutions	COPS	Any
User interface	API	?	Others	API

Table 1: Mapping of the proposals into the categories.

In Table 1, a mapping of the proposals into the described categories is depicted. Some comments on this table follow. First of all, it has been considered that each proposal has been aimed at the Data Grid application model if it only includes support for the network resource. This decision was made because network is specially significant for that kind of applications. Proposals which include support for a variety of resources are not considered to be aimed at any application model in particular, as they can deal with the requirements of any application (requirements for CPU, network or storage).

With regard to the information service each proposal uses, GARA uses MDS [17], and G-QoSM uses UDDIe [7]. In reference to NRSE and GNB, no specific information service has been found, so they are considered to support any information service.

Concerning the scheduling method, which consists in mapping jobs to resources, some proposals do not provide this feature. Only GARA, G-QoSM and GNB provide scheduling. The algorithms used in GARA for scheduling are DSRT [14] and PBS [28], whilst in G-QoSM is DSRT. These algorithms only pay attention to the load of the computing resource, thus a powerful unloaded computing resource with a overloaded network can be chosen to run jobs, which decreases the performance received by users, especially when the job requires a heavy network input/output.

As for the scheduling target, it is assumed that GARA and G-QoSM are aimed at minimizing the application deadline, as they try to schedule each job to a computing resource with as much free CPU as possible. On the other hand, GNB tries to minimize both the application deadline, and also the loss of data through the network path input/output files will follow.

About inter-domain protocols, the authors have not been able to find GARA's approach for this. But, as a GARA server is required in each administrative domain that is traversed, it is assumed that GARA uses a set of specific protocols, not standard protocols.

As for intra-domain protocol, GARA and NRSE use proprietary solutions. This means that, as they have been built to work with Cisco routers, they issue Cisco-IOS commands to configure routers.

In relation to user interfaces, a web interface has been implemented for GARA [5], although it does not belong to its core. Also, G-QoSM provides a number of user interfaces [37].

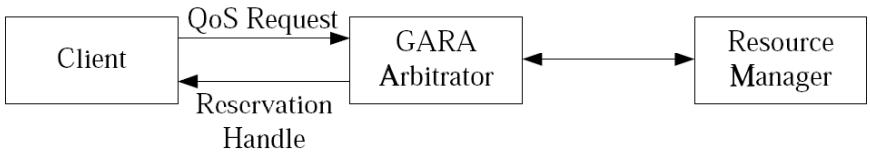

Figure 6: GARA system [33].

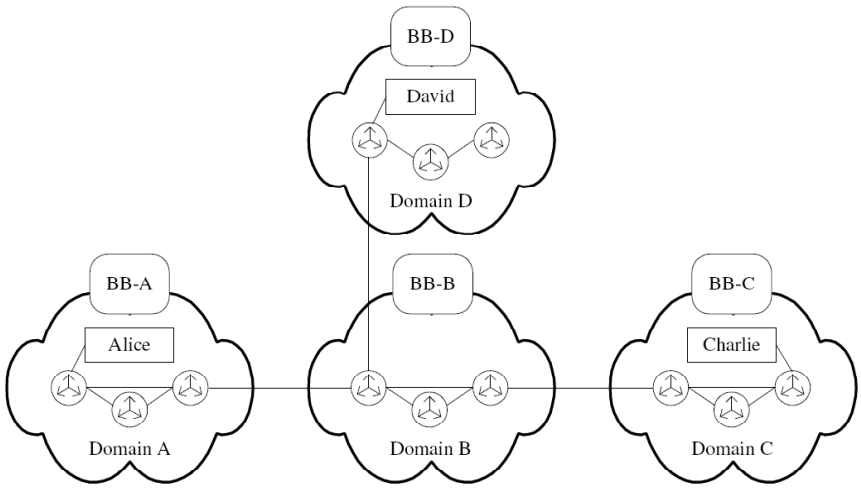

Figure 7: GARA multidomain reservations [33].

3.1 General-purpose Architecture for Reservation and Allocation (GARA)

General-purpose Architecture for Reservation and Allocation (GARA [33]) provides programmers and users with convenient access to end-to-end quality of service (QoS) for computer applications. It provides uniform mechanisms for making QoS reservations for different types of resources, including computers, networks, and disks. These uniform mechanisms are integrated into a modular structure that permits the development of a range of high-level services.

GARA is a straightforward system, as illustrated in Figure 6 [33]. A GARA-enabled program makes a request for a QoS reservation to the GARA Arbitrator. This request is specified in a uniform way: requests are not substantially different for differ-

ent types of resources and it is easy to make many such requests. The GARA Arbitrator communicates with a resource manager. Resource managers may be part of the GARA system, or may be provided by other systems: the GARA Arbitrator ensures seamless communication in either case. The resource manager decides if a QoS reservation can be granted or not. Assuming it can be granted, the GARA Arbitrator returns an opaque string, called a reservation handle, to the requesting application. This reservation handle can be used to manipulate the reservation by modifying, canceling, or querying it.

In addition to this basic interaction, GARA provides asynchronous feedback to programs. That is, when there are changes to a reservation, programs can be immediately informed of the change. Feedback ranges from notification that a reservation has begun or ended to notification that a reservation is apparently too small for the application's needs.

The main features of GARA are four. First, it provides a way to create immediate and advanced reservations for a diverse set of resources. Second, because of that, it is claimed that it is easy to build higher-level services on top of GARA. Third, it is also claimed that the layered architecture of GARA allows for easy extension as new resources become available. Fourth, GARA uses a security infrastructure, so that requests are authenticated and authorized.

With regard to multidomain reservations spanning several domains, the user (or a broker acting in his behalf) has to authenticate into all the domains. More accurately, the user has to authenticate with the bandwidth broker of all each of the domains. This makes GARA difficult to scale. Apart from that, GARA may have interference problems between users, as depicted in Figure 7 [34]. In this figure, an user, Alice, wants to send traffic from her machine to Charles' machine. So, she needs a reservation in the domains A, B and C. Another user, David, who also wants to send some traffic to

Charles' machine, has reserved bandwidth only in domains D and B, but not in C. As domain C policies traffic based on traffic aggregates, not individual users, it can not tell the difference between Alice reserved traffic and David's traffic. So, there will be more reserved traffic entering domain C than C expects, which will make C drop or downgrade the extra traffic, affecting Alice's reservation.

But GARA has more limitations. Some resources like disk space are fundamentally very different from network capacity. These resources are localized to certain end-systems and reservations can be made at the remote end-systems where such resources are located. Network capacity is a distributed resource requiring reservations at the local and remote end-systems as well as the network path between the local and remote systems. Regarding to multidomain reservations, GARA must exist in all the traversed domains, and the user (or a broker acting in his behalf) has to authenticate into all the domains. This makes GARA difficult to scale.

3.2 Network Resource Scheduling Entity (NRSE)

Other well-known proposal is Network Resource Scheduling Entity (NRSE [8]). Its authors say that signalling and per-flow state overhead can cause end-to-end QoS reservation schemes to scale poorly to large numbers of users and multi-domain operation, as could be seen when using INTSERV and RSVP and also with GARA [8]. This has been addressed by them by storing the per-flow/per application state only at the end-sites that are involved in the communication. Service requests can be dynamic or can be in advance.

NRSE has a similar role to GARA, but without GARA's limitations. The NRSE is able to automatically negotiate a multi-domain reservation by communicating with

its counterpart on the remote network, on behalf of its client. Thus the system is highly scalable.

The main features of the NRSE are (in no particular order) [8]:

- Allows reservations of network capacity across multiple domains based on SLAs
- Allows decentralized state for the reservations and does not require NRSEs to hold reservation state at all domain boundaries, just at end-sites which are involved in the reservation
- Allows creation, deletion and modification of reservations
- Allows configurable notifications from NRSEs to applications regarding violations to QoS
- The flow identification can use most of the standard IP, TCP and UDP header fields
- QoS service class, directionality and policing can be specified in the SLA
- The NRSE uses a localized polling mechanism for the application holding the reservation (keepalive) so that resources can be reclaimed if an application fails
- Allows flexible scheduling of jobs to deadlines for non-real-time service-requests (e.g. file transfers)
- Can be configured with local policies that control the operation of the system, with such local policies being autonomously managed
- Has a hierarchical trust model so that security and access control information remains as localized as possible

Although NRSE has demonstrated its effectiveness in providing DiffServ [9] QoS, it is not clear how a Grid application developer would make use of this capability especially as the application programming interface is not clearly defined [6].

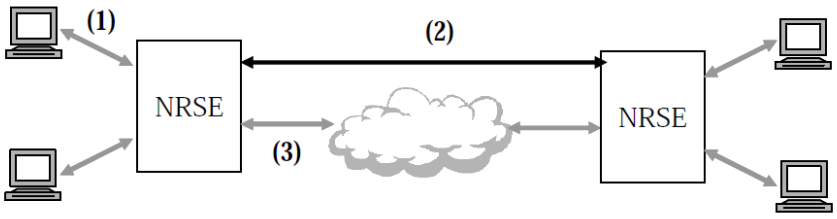

Figure 8: The NRSE reservation model [8].

3.3 Grid Quality of Service Management (G-QoSM)

Grid Quality of Service Management (G-QoSM [6]) is a framework to support QoS management in computational Grids in the context of the Open Grid Service Architecture (OGSA). G-QoSM is a generic modular system that, conceptually, supports various types of resource QoS, such as computation, network and disk storage.

G-QoSM has three main operational phases: establishment, activity and termination. During the establishment phase, a client application states the desired service and QoS requirements. G-QoSM then undertakes a service discovery, based on the specified QoS properties, and negotiates an agreement for the client application. During the activity phase additional operations, such as QoS monitoring, adaptation, accounting, and possibly re-negotiation may take place. During the termination phase the QoS session is ended due to an expiring resource reservation, an agreement being discovery, based on the specified QoS properties, and negotiates an agreement for the client application. During the activity phase additional operations, such as QoS monitoring, adaptation, accounting, and possibly re-negotiation may take place. During the ter-

mination phase the QoS session is ended due to an expiring resource reservation, an agreement being violated or a service completion; resources are then freed for use by other clients.

This framework aims to provide three main functions [6]: 1) support for resource and service discovery based on QoS properties; 2) provision for QoS guarantees at application, middleware and network levels, and the establishment of SLAs to enforce QoS parameters; and 3) support for QoS management of allocated resources, on three QoS levels: 'guaranteed', 'controlled load' and 'best effort'. G-QoSM also supports adaptation strategies to share resource capacity between these three user categories.

3.4 Grid Network Broker (GNB)

The author claim that communication between users and resources involved in a Grid is carried out through network links, hence network links are an essential part of a Grid system. Because of this, they should be considered not only as a resource to be scheduled, but also as a parameter to decide about the convenience of using other different resources, such as CPU. This is, if the network links of a computing resource are overloaded, then this fact should affect whether this computing resource is chosen to run a user's job.

All of this is achieved by means of a network-aware grid meta-scheduler, also known as *Grid Network Broker*, or *GNB* [11] [12]. This entity performs meta-scheduling of users' jobs to resources, and this task is done taking into account the quality of the network links and their level of utilization. Apart from that, GNB schedules link bandwidth, in order to keep the network away from saturation.

The architecture of the GNB is shown in Figure 9. It includes the following modules:

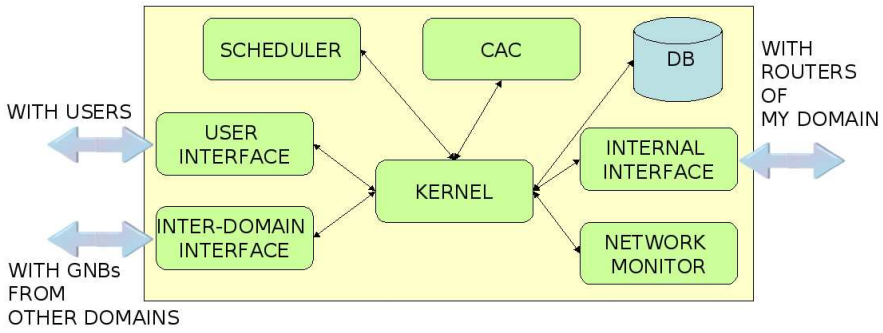

Figure 9: Architecture of the GNB [11].

User interface: This module will allow users to submit requests to the GNB. More precisely, a user can ask the GNB to allocate a computing resource to run his/her jobs. As a consequence, the 'Scheduler' module will order the available and suitable resources from the best to the worst, and choose the best of them. Then, the 'CAC' module will perform the connection admission control test.

Scheduler: This is the main module of the GNB, as it carries out the meta-scheduling of jobs to computing resources. As mentioned above, this task is done taking into account the network as a key parameter. Thus, network links will have the importance they deserve in the Grid.

This module selects the resource where every job will be run, attending to different criteria. In order to do this, the 'Scheduler' needs to retrieve a list of the available resources from the GIS module.

Several scheduling algorithms can be implemented, and the one most suitable applied to each job depending on its profile. The *profile* of a job is a high-level definition of its requirements (i.e., compute-intensive, interactive, or data-intensive job). The profile of the job is a hint to the scheduler so that it can apply the most suitable selection criteria. For example, if a job is compute-intensive, the scheduler will make its selection with the parameters related to the resource computing power as the most

importants. On the other hand, if the resource is data-intensive, the scheduler will regard the network related parameters as the most important ones.

Connection admission control (CAC): This module checks the links between an user and a given resource, previously selected by the 'Scheduler' module. The CAC assess whether a new connection can be admitted through every link of the network path between the user and the resource. That is, it will check if the links in the path from the user's node to the resource have enough capacity to support the new connection, without degrading the QoS of the ongoing connections. The CAC module will need the information stored in the database regarding the ongoing connections, i. e., the path they traverse and their allocated bandwidth. It will also need network monitoring information to make an estimation on the amount of background traffic, which is transmitted without previous reservation. The CAC module should support both immediate and advanced reservations.

Interdomain interface: A GNB communicates with GNBs from other neighbor administrative domains in order to reserve computing resources in different domains. A GNB needs this communication whenever the destination of the user's flow is outside its domain. This module also has to deal with Service Level Agreements (SLAs) among neighboring domains. The SIBBS (Simple Interdomain Bandwidth Broker Signalling) protocol [32] is frequently used for this task.

Internal interface: The GNB needs communication with edge routers as well as with the core routers within the domain. The routers have the ability of handling the application traffic. On the other hand, the GNB keeps a complete view of the topology of the network and manages the information about the established SLAs with other domains. Thus, the GNB configures the routers of the domain according to this information.

The GNBs needs to retrieve topology information and to configure the routers following adequate policy decisions in order to provide network QoS. Routing tables are retrieved just by listening to the appropriate routing messages, as pointed out above. On the other hand, in order to modify router configuration, this interface can use the COPS [16] protocol, send SNMP updates, or even opening a TELNET session and issuing router configuration commands. The use of standard protocols should always be preferred, for the sake of interoperability. However, a careful modular design can make straightforward the addition of specific configuration information for particular router brands.

Database: All the information the GNB manages and uses must be stored in a database. Besides the network topology data and the information on the established SLAs, which have been mentioned above, dynamic information on network conditions must be stored too, and updated frequently. Examples of such dynamic data include estimations on available network link bandwidth and latencies, statistics on link failure, and information on ongoing reservations. The GNB will take into account these data in order to select an adequate resource to run a job. Dynamic data will be collected by the *Network Monitor* module, which will be described the next.

Network monitor: The purpose of this module is to run tests on the network so that information on current load and delay conditions can be retrieved and stored in the 'Database module'. These tests are usually based on sending probes through different paths in the network (as in the Network Weather Service [43]), thus adding traffic overhead. The more frequently tests are run, the more accurate estimations on network performance are, but the more overhead is caused. Thus, this is an extremely sensitive issue because a trade-off must be achieved between overhead and accuracy on network status.

Another approach for gathering information on link usage could be through the use of the SNMP and RMON management and monitoring protocols. A Management Information Base (MIB) is maintained in every SNMP-compatible router, which holds information on the number of bytes or packets transmitted per interface. More over, the RMON monitoring protocol can be configured to fire alarms when the amount of data transmitted through a given interface is over or under certain threshold values. If these alarms (which can generate SNMP traps) are received by the GNB, it can make out an estimation on the level of use of every router interface (i.e. link) in the network. Obviously, security concerns prevent from extending this approach outside an administrative domain. Thus, a GNB per administrative domain should exist, and some criteria for the amount and detail of the information shared among neighboring GNBs should be devised.

4 Conclusion

This paper has presented a taxonomy of network QoS proposals for Grid systems, and applied the taxonomy on the main existing proposals. The taxonomy comprises the application model each proposal has been aimed at, the information service it uses, types of resources each proposal can deal with, the method each proposal uses to provide scheduling (for those proposals which provide scheduling), and communication protocols.

A number of proposals have been surveyed using the taxonomy. This paper thus helps to identify some approaches for designing network QoS proposals for Grid systems.

The most well-known proposal is GARA, which is aimed at providing QoS over a variety of resources. But GARA is not the only one. Other proposals are NRSE, G-

QoSM, and GNB. The main advantages and disadvantages of them have been presented here.

Some of the proposals provide QoS on a variety of resources, whilst others only provide network QoS. Some of them provide immediate and advance reservations, whilst other only provide immediate reservations. Some of them provide scheduling of jobs into computing resources, by means of different algorithms.

The aim of all the proposals is to provide users with a way to request network reservations, so that network QoS can be guaranteed. Moreover, the aim of GNB is including the network as a decision parameter when choosing a computing resource for a given job. This is, not only do its authors want to be able to reserve bandwidth, but also they want the network to be a more active part.

Apart of the proposals which have been surveyed here, another proposal whose aim is the provision of network QoS in Grid systems is GNRB [4]. This proposals has not been included in this taxonomy because very little information has been found about it.

Acknowledgement

This work has been jointly supported by the Spanish MEC and European Comission FEDER funds under grants "Consolider Ingenio-2010 CSD2006-00046" and "TIN2006-15516-C04-02"; by JCCM under grants PBC-05-007-01, PBC-05-005-01 and José Castillejo.

References

[1] GRIST: Grid Data Mining for Astronomy. Web Page, 2007. `http://grist.`

caltech.edu/.

[2] LCG (LHC Computing Grid) Project. Web Page, 2007. http://lcg.web.cern.ch/LCG.

[3] Abramson, Giddy, and Kotler. High performance parametric modeling with nimrod/G: Killer application for the global grid? In *IPPS: 14th International Parallel Processing Symposium* (2000), IEEE Computer Society Press.

[4] Adami, D., Carlotti, N., Giordano, S., and Repeti, M. *Distributed Cooperative Laboratories: Networking, Instrumentation, and Measurements*. Springer Berlin Heidelberg, 2006, ch. Design And Development Of A GNRB For The Coordinated Use Of Network Resources In A High Performance Grid Environment.

[5] Adamson, W. A., and Kornievskaia, O. A practical distributed authorization system for GARA. In *InfraSec* (2002), G. I. Davida, Y. Frankel, and O. Rees, Eds., vol. 2437 of *Lecture Notes in Computer Science*, Springer, pp. 314–324.

[6] Al-Ali, R., et al. Network QoS Provision for Distributed Grid Applications. In *Proc. of the Intl. Journal of Simulations Systems, Science and Technology* (2004).

[7] Ali, A. S., Rana, O. F., Al-Ali, R. J., and Walker, D. W. UDDIe: An extended registry for web service. In *SAINT Workshops* (2003), IEEE Computer Society, pp. 85–89.

[8] Bhatti, S., Sørensen, S., Clark, P., and Crowcroft, J. Network QoS for Grid Systems. *The Intl. Journal of High Performance Computing Applications 17*, 3 (2003).

[9] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and Weiss, W. Internet RFC 2475 : An architecture for differentiated services., 1998.

[10] Buyya, R., Abramson, D., and Giddy, J. Nimrod/G: An architecture of a resource management and scheduling system in a global computational grid. *CoRR cs.DC/0009021* (2000).

[11] Caminero, A., Carrion, C., and Caminero, B. On the improvement of the network QoS in a grid environment. In *Proceedings of the 4th Intl. Workshop on Middleware for Grid Computing - MGC 2006* (2006), ACM.

[12] Caminero, A., Carrion, C., and Caminero, B. Designing an entity to provide network QoS in a grid system. In *Proceedings of the 1st Iberian Grid Infrastructure Conference, IberGrid 2007* (2007).

[13] Chervenak, A., Foster, I., Kesselman, C., Salisbury, C., and Tuecke, S. The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets, Mar. 02 1999.

[14] Chu, H.-H., and Nahrstedt, K. CPU service classes for multimedia applications. In *ICMCS, Vol. 1* (1999), pp. 296–301.

[15] Corbatto, M. An introduction to portable batch system (PBS). Web Page, 2007. `http://hpc.sissa.it/pbs/pbs.html`.

[16] Durham, D., Boyle, J., Cohen, R., Herzog, S., Rajan, R., , and Sastry, A. The COPS (Common Open Policy Service) Protocol. RFC 2748, January 2000.

[17] Fitzgerald, S., Foster, I., Kesselman, C., von Laszewski, G., Smith, W., and Tuecke, S. A directory service for configuring high-performance distributed computations. In *Proc. 6th IEEE Symp. on High Performance Distributed Computing* (1997), IEEE Computer Society Press, pp. 365–375.

[18] Foster, I. What is the Grid? A Three Point Checklist. *Grid Today* (July 2002).

[19] Foster, I., and Kesselman, C. *The Grid 2: Blueprint for a New Computing Infrastructure*, 2 ed. Morgan Kaufmann, 2003.

[20] Foster, I., Kesselman, C., Nick, J., and Tuecke, S. *Grid Computing: Making the Global Infrastructure a Reality.* Wiley InterScience, 2003, ch. The Physiology of the Grid.

[21] Foster, I. T. The anatomy of the Grid: Enabling scalable virtual organizations. In *CCGRID* (2001), IEEE Computer Society, pp. 6–7.

[22] Hayashibara, N., Cherif, A., and Katayama, T. Failure detectors for large-scale distributed systems. In *The 21th IEEE Symp. on Reliable Distributed Systems, (SRDS '02)* (Washington - Brussels - Tokyo, Oct. 2002), IEEE, pp. 404–409.

[23] Henricksen, K., and Indulska, J. Adapting the web interface: An adaptive web browser, 2001.

[24] Hoschek, W., Jaén-Martínez, F. J., Samar, A., Stockinger, H., and Stockinger, K. Data management in an international Data Grid project. In *GRID* (2000), R. Buyya and M. Baker, Eds., vol. 1971 of *LNCS*, Springer, pp. 77–90.

[25] Huedo, E., Montero, R. S., and Llorente, I. M. A framework for adaptive execution in grids. *Softw. Pract. Exper. 34*, 7 (2004), 631–651.

[26] Jha, S., and Sohail, S. The Survey of Bandwidth Broker. Tech. rep., School of Computer Science and Engineering, The University of New South Wales, 2002.

[27] Krauter, K., Buyya, R., and Maheswaran, M. A taxonomy and survey of grid resource management systems for distributed computing. *Softw, Pract. Exper 32*, 2 (2002), 135–164.

[28] Mateescu, G. Extending the Portable Batch System with preemptive job scheduling. In *SC2000: High Performance Networking and Computing. Dallas Convention Center, Dallas, TX, USA, November 4–10, 2000* (2000), ACM, Ed., ACM Press and IEEE Computer Society Press, pp. 148–148.

[29] Mendes, P., and Nichols, K. Requirements for DiffServ Control Plane Elements. January, 2006. Draft document available at `http://tools.ietf.org/id/draft-mendes-dcpel-requirements-00.txt`.

[30] Nahrstedt, K., hua Chu, H., and Narayan, S. QoS-aware resource management for distributed multimedia applications. Tech. Rep. 2030, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois, 1997.

[31] Plale, B., Dinda, P., Helm, M., and von Laszewski, G. Key concepts and services of a grid information service. In *Proc. of the 15th Intl. Conference on Parallel and Distributed Computing Systems (PDCS 2002)* (2002).

[32] QBone Signalling Design Team. Simple Interdomain Bandwidth Broker Signalling (SIBBS). `http://qbone.internet2.edu/bb/`, 2002.

[33] Roy, A. *End-to-End Quality of Service for High-End Applications*. PhD thesis, Dept. of Computer Science, University of Chicago, 2001.

[34] Roy, A., and Sander, V. *Grid Resource Management*. Kluwer Academic Publishers, 2003, ch. GARA: A Uniform Quality of Service Architecture, pp. 377–394.

[35] Seppanen, P., and Paivike, H. A time-driven extension for a commercial multi-purpose operating system seppanen. In *Proceedings of the Fifth Euromicro Workshop on Real-Time Systems* (1993).

[36] Smith, W., Foster, I., and Taylor, V. Scheduling with advanced reservations. In *14th International Parallel and Distributed Processing Symposium (SPDP'2000)* (Washington - Brussels - Tokyo, May 2000), IEEE, pp. 127–132.

[37] Sohail, S., Pham, K. B., Nguyen, R., and Jha, S. Bandwidth Broker Implementation: Circa-Complete and Integrable. Tech. rep., School of Computer Science and Engineering, The University of New South Wales, 2003.

[38] Stelling, P., DeMatteis, C., Foster, I. T., Kesselman, C., Lee, C. A., and von Laszewski, G. A fault detection service for wide area distributed computations. *Cluster Computing 2*, 2 (1999), 117–128.

[39] Sulistio, A., and Buyya, R. A grid simulation infrastructure supporting advance reservation. In *16th International Conference on Parallel and Distributed Computing and Systems, Cambridge, USA, November 9–11 2004.* (2004).

[40] The Portable Batch System. Web Page, 2007. `http://www.nas.nasa.gov/Software/PBS/pbsnashome.html`.

[41] Venugopal, S. *Scheduling distributed data-intensive applications in global grids.* PhD thesis, Department of Computer Science and Software Engineering, The University of Melbourne, 2006.

[42] Venugopal, S., Buyya, R., and Winton, L. J. A grid service broker for scheduling distributed data-oriented applications on global grids. In *Proceedings of the 2th International Workshop on Middleware for Grid Computing - MGC 2004* (2004), ACM.

[43] Wolski, R., Spring, N. T., and Hayes, J. The network weather service: A distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems 15*, 5–6 (1999), 757–768.